CMR-1309 Project Plan for Search Relevancy Improvements

Link to white paper: ImprovingSearchRelevanceintheCommonMetadataRepository.docx

1. Develop the *gold standard* test set

This test set will be used to evaluate the performance of the CMR with respect to relevancy.

1.1 Take a snapshot of the current production collections. - 2h

This can initially consist of an actually copy of the elastic index, but will eventually need to be in a more compact form that can be used in testing without requiring Elasticsearch.

1.2 Create a set of queries that cover the information space of the collections, i.e., queries that when combined together would pull back all the collections. - 4h

1.3 Identify the collections that should be returned by each query in the set - preferably ranked in order of most relevant to least. - 1.5D

This is likely to be an iterative process and will require cooperation from the subject experts, e.g., DAACs.

2. Compute baseline metrics

2.1 Write code to compute metrics given the gold standard set and an actual search result set using the metrics discussed in the white paper or other metrics as deemed appropriate. - 14D

2.1.1 Precision & Recall - 2D

2.1.1 F-Measure & Weighted F-Measure - 1D

2.1.2 Area Under Curve (AUC) - 3D

2.1.2.1 AUC numeric calculation - 1D

2.1.2.2 AUC plot - 2D

2.1.3 AP Correlation - 3D

2.1.4 Rank Biased Overlap - 3D

This code should be implemented as a library that can be re-used in the sand box application.

2.2 Execute the gold standard queries against the collections snapshot. - 1D

2.2.1 Store the results as the *baseline results*.

2.3 Use the baseline results, gold standard, and code from 2.1 to compute the *baseline metrics*. - 0.5D

2.3.1 Store the baseline metrics with the baseline results

3. Develop the sandbox application

3.1 Design Java interface with an API to allow scoring of a collection model (UMM?) for a given query - 2D

3.1.1 Scores should be a float value between 0 and 1.

3.2 Design and implement an application to read a config file, then use that config to load implementations of the interface in 3.1 for various heuristics - 7D

3.2.1 Design - 2D

3.3.1 Implementation - 5D

3.3 Application should download gold standard set and sort it using weighted combination of scoring implementations from 3.2

3.4 Application should compute metrics using the code from 2.1

4. Implement *Data Content Search-Keyword heuristic* as described by Chris Lynnes in sandbox and compute metrics

TODO - break down and estimate these

5. Implement *Temporal and Spatial Coverage* heuristic as described by Chris Lynnes in sandbox and compute metrics

- Need more detail about scoring - does a collection that more closes covers the search area get a higher relevance than a much larger one that intersects the same area?

- TODO - break these into two different heuristics and then break these down and estimate

- Can we re-use CMR code here?

6. Prototype

6.1 CMR Elasticsearch plugin for heuristics

Plugin should reuse code from sandbox application and heuristics implementations - 11D

6.1.1 Setup plugin - 1.5D

6.1.2 Implementation -

6.1.2.1 Reimplement keyword heuristic for elastic plugin - 2D

6.1.2.2 Reimplement temporal heuristic for elastic plugin - 1D

7. Test prototype with heuristics in the workload environment - 2.5D

7.1 Collect query performance (speed) metrics. - 2D

7.2 Compute impact of changes on query times (difference between baseline and post change to support heuristics) - 0.5

7.2.1 Verify query times remain sub-second to the 99th percentile.

Assumptions/Questions

- Only search relevance for collection queries will be addressed in this approach.
- Stakeholders will participate in creating the gold standard test set.
- Heuristics can be added without significantly affecting query times.
- None of the changes considered here have a significant impact on ingest times.
- Do we want to maintain the sandbox past the prototype phase. Specifically should there be code sharing between the elastic search plugin and the heuristics implementations so that the sandbox can be used for future testing and improvements to search relevancy. This impacts the Elasticsearch implementation estimate.
- Can the sandbox code re-use CMR code?
- Will the sandbox code be shared through the ECC?
- Who develops the heuristic implementations? Do we legitimately expect outside parties to do this?
- Is it realistic that we are going to get participation from DAACs in generating the gold standard?
    - There is a significant risk to completion and improving results if we do not.
- When must this be in production? What is the end result of the task we are estimating now? Is the goal to have the Elasticsearch plugin in production at the end of this task or is it to have implemented and verified heuristics (improvements in relevancy)?

Additional thoughts:

The heuristics as described by Chris Lynnes can only sort collections that would already be returned by Elasticsearch. As such, if collections that should be returned by a query are not, the heuristics will not help. They *can help* to move relevant results to the front, thereby increasing apparent recall, but they can't add things in that aren't there.

So, if we need to improve recall, we will have to look to other methods (adding additional fields to the aggregate keyword field, synonym expansion, etc.).

Time permitting, we should implement heuristic sets as described in the white paper.